Arkena

U S E R G U I D E

Testgrid API Documentation

Arkena
27, bld Hippolyte Marquès
94200 Ivry sur Seine
FRANCE

Last Update: 08.2014

Table of Contents

Arkena

Testgrid

You can checkout the code at this address: git@github.com:SmartJog/Testgrid.git

Client Instanciation

```
user = testgrid.client.User("toto")
client = testgrid.rest.client(user = user, host = "127.0.0.1:3000")
```

Session creation

```
session = client.open_session("test")
```

Deploy Package

Getting a package from a specific type.

Installing all the package in nodes assigned to a specific session.

```
package = client.get_package("DebianPackage", pkg)
plans = session.deploy(package)
for pkg, node in plan:
        print "package", pkg , "installed on node", node
```

Arkena

Undeploy a specific session

Uninstall and release all the nodes in a specific session.

```
session.undeploy()
```

List node

```
for node in client.get_nodes():
  print node.name
```

Allocation

Allocate a node according the required options.

```
_opts = { “sysname” : “wheezy64” }
node = session.allocate_node(**_opts)
```

Release node

```
node = client.get_node(name)
session.release(node)
```

Arkena

Node methods

```
pkg = testgrid.debian.Package(name = "fleche", version = "1.0")
if node.is_installed(pkg):
        node.uninstall(pkg)
elif  node.is_installable(pkg):
        node.install(pkg)
```

Ansible integration

Run ansible-playbook through Testgrid

```
session = client.open_session(name)
playbook = testgrid.anspkg.Playbook(pkg_name = "motherbrain", session)
playbook.run()
```

Implementation of testgrid.model

Grid Implementation

There are various type of grid:

- Persistent grid (grid that store all Testgrid data in sqlite3 database)
- Vagrant grid (Inherite from Persistent grid. Generate Vagrant nodes).

For example, the module testgrid.vgadapter.grid creates Vagrant nodes.

```
def _create_node(self, pkg = None, **opts):
        """
        Override to create a new node (hereafter called a "transient node")
```

```
        *  compatible with package $pkg
        *  supporting specified options $opts
        Warning: do not call this directly; it's invoked by find_node on demand.
        """
```

Node Implementation

In order to implement your own Node, you can override those abstract methods.

```
class Node(object):
        "a node abstracts any object supporting packages & services"

        __metaclass__ = abc.ABCMeta

        def __init__(self, name):
             self.name = name

        def __str__(self):
             return self.name

        def get_info(self):
             return "no details"

        @abc.abstractmethod
        def has_support(self, **opts):
             "return True if all specified options are supported"
             pass

        @abc.abstractmethod
        def get_load(self):
             "return a float as a composition of load measures"
             pass

        @abc.abstractmethod
```

```
def join(self, subnet):
     "setup a network interface on the specified subnet"
     pass

@abc.abstractmethod
def leave(self, subnet):
     "remove the network interface on the specified subnet"
     pass

@abc.abstractmethod
def get_subnets(self):
     "return the list of subnets the node belongs to"
     pass

@abc.abstractmethod
def get_hoststring(self):
    "return node hoststring as [user[:pass]@]hostname[:port]"

@abc.abstractmethod
def get_installed_packages(self):
    " get installed package on the node"
```

Package Implementation

In order to implement your own package, you can override those abstract methods.

```
@abc.abstractmethod
def install(self, node):
       "install package on $node, raise exception on error"
       pass

@abc.abstractmethod
def uninstall(self, node):
```

```
                "uninstall package from $node, raise exception on error"
                pass

        @abc.abstractmethod
        def is_installed(self, node):
                "return True if the package is installed on $node, False otherwise"
                pass

        @abc.abstractmethod
        def is_installable(self, node):
```

Example Implementation Package Debian

```
class Package(model.Package):
        "debian package management commands"

        def __init__(self, *args, **kwargs):
                super(Package, self).__init__(*args, **kwargs)
                if self.version:
                        self.tag = "%s=%s" % (self.name, self.version)
                else:
                        self.tag = self.name

        def install(self, node):
                return self._run_apt(node = node, state = "present force=yes")

        def uninstall(self, node):
                return self._run_apt(node = node, state = "absent purge=yes force=yes")

        def is_installed(self, node):
                code, stdout, stderr = node.execute("dpkg -s %s" % self.name)
                return code == 0
```